Vysoká škola báňská - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Absolvování individuální odborné praxe
Individual Professional Practice in the Company

2014 Bušfy Patrik

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Patrik Bušfy**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Absolvování individuální odborné praxe
Individual Professional Practice in the Company

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě: VÍTKOVICE IT SOLUTIONS a.s.
2. Struktura závěrečné zprávy:
a) Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta
b) Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti
c) Zvolený postup řešení zadaných úkolů
d) Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe
e) Znalosti či dovednosti scházející studentovi v průběhu odborné praxe
f) Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení

Seznam doporučené odborné literatury:

Podle pokynů konzultanta, který vedl odbornou praxi studenta.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Pavel Moravec, Ph.D.**

Konzultant bakalářské práce: Mgr. Jiří Chamrád

Datum zadání: 01.09.2013

Datum odevzdání: 07.05.2014

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Čestné prohlášení studenta

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně, pod odborným vedením vedoucího bakalářské práce a používal jsem literaturu uvedenou v práci, ze které jsem čerpal.

V Ostravě dne: *5. 5. 2014*

Podpis: .....

Prohlášení firmy

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.

V Ostravě dne: *5. 5. 2014*

Podpis: .....

VÍTKOVICE
VÍTKOVICE IT SOLUTIONS a.s.
Cihelní 1575/14
702 00 Ostrava, Moravská Ostrava
IČ 28606582
DIČ CZ28606582

Poděkování

Děkuji vedoucímu bakalářské práce Ing. Pavel Moravec, Ph.D. a kolektivu firmy VÍTKOVICE IT SOLUTIONS a.s. za cenné rady, připomínky a metodické vedení práce.

Abstrakt

Bakalářská práce pojednává o absolvování praxe v externí společnosti VÍTKOVICE IT SOLUTIONS a.s. Ostrava. Hlavní náplní bakalářské práce je popis zadaných úkolů a jejich praktické řešení s ukázkou kódu pomocí implementace jazyka C#. Práce také zahrnuje popis dvou softwarových projektů řešených po celou dobu absolvování odborné praxe. Projekty zahrnují práci s informačním systémem, implementace konstrukčních kusovníků a přepočty normo časů. V závěru bakalářské práce jsou zhodnoceny získané praktické zkušenosti a dojmy z odborné praxe.

Klíčová slova

C#, C++, Konstrukční kusovník, Hardis, Odborná praxe, Bakalářská práce, Databáze, .NET.

Abstract

This thesis describes professional work experience in VÍTKOVICE IT SOLUTIONS a.s. Ostrava company. The main part of the thesis is the description of assigned tasks and their practical solutions with C# code examples. The work also includes a description of two software projects in which have been solved during the profesional practice. The projects include the work with the information system, implementation of engeneering bills of design materials and conversions of standart times. In conclusion of the thesis, there is an evaluation of gained experience and overall impressions of the professional practice.

Key words

C#, C++, Engeneering bills of design materials, Hardis, Professional practice, Bachelor thesis, Database, .NET.

Seznam použitých zkratek

Zkratka	Význam
BG	BoomGrid
C#	C Sharp
CAD	Computer-aided drafting
ERP	Plánování podnikových zdrojů
IKK	Integrovaný Konstrukční Kusovník
IS	Informační systém
MPSV	Ministerstvo práce a sociálních věd
PLM	Správa životního cyklu výrobku
TFS	Team Foundation Server
TG	TabGrid
VPN	Virtuální privátní síť
WF	Windows Forms

Obsah

1. Úvod

Bakalářská práce formou absolvování odborné praxe má seznámit čtenáře s náplní mé práce ve společnosti VÍTKOVICE IT SOLUTIONS a.s. Během stáže jsem měl za úkol vypracovat řadu úkolů v softwaru pro vkládání konstrukčních kusovníků do databáze, dále pomoci již ve funkčním projektu Hardis, kde byly odhaleny nedostatky při testování a ty bylo potřeba změnit nebo vylepšit. Na obou projektech jsem spolupracoval s přiděleným konzultantem panem Mgr. Chamrádem. Pouze v projektu s konstrukčním kusovníkem mi byly úkony zadávány konzultantem. V případě projektu Hardis jsem byl začleněn do většího týmu, kde jsem dostával pokyny od pana Vengela. Po skončení stáže budou projekty postupně dopracovány.

Na počátku této práce naleznete stručný popis toho, co mne motivovalo vybrat si právě bakalářskou práci formou praxe. Následně obsahuje seznámení se společností VÍTKOVICE IT SOLUTIONS a.s. Práce je rozdělena do dvou celků. Tyto celky obsahují popisy dvou projektů, na kterých jsem pracoval. Nejprve od jejich popisu po následné ukázky kódů s principem funkčnosti. V závěru celé práce můžeme nalézt porovnání nabytých zkušeností při absolvování praxe s vědomostmi ze studia na vysoké škole.

1.1 Příprava na praxi

Odbornou praxi jsem si začal zajišťovat v dostatečném časovém předstihu. Nejprve jsem měl domluvenou odbornou praxi u podnikatele v místě svého bydliště. Jeho firma se zabývá instalací softwaru do počítačů, odvirování počítačů, tvorbou webových stránek a hlavně poskytuje připojení k internetu. Nezaobírá se vývojem softwaru, proto jsem nakonec tuto možnost nevyužil. Začal jsem vyhledávat v seznamu, který jsem obdržel od katedry informatiky. Vybíral jsem firmu, která má zaměření na vývoj softwaru v programovacím jazyku C#, kterému bych se rád věnoval ve svém budoucím povolání. Firmy se zaměřením na vývoj softwaru jsem si vybral tři a absolvoval jsem pouze jeden pohovor - a to ve firmě VÍTKOVICE IT SOLUTIONS a.s., která sídlí v Ostravě – Dolní oblast Vítkovice.

Pohovor u této firmy se skládal ze dvou částí. První část pohovoru byla pouze informativní. Dotazovali se na programovací jazyk, který mi nejvíce vyhovuje a jaké mám zkušenosti v oboru. Druhá část pohovoru se konala o týden později, zde mi byl vysvětlen informační systém, na kterém bych měl v průběhu absolvování praxe pracovat. Následně mi

byla zadána odborná úloha pro domácí vypracování. Úkolem bylo vytvořit návrh pro vkládání kusovníku do databáze, který obsahuje různé podsestavy a součásti. Na vypracování tohoto úkolu jsem dostal týden. Po týdnu jsem jim práci odeslal e-mailem. Následně mi přišel z firmy potvrzující email, že jsou s prací spokojeni a na základě mého úkolu mě přijali. Termín nástupu na odbornou praxi byl domluven od 1. 10. 2013 v rozsahu 50 dní.

1.2 O společnosti VÍTKOVICE IT SOLUTIONS a.s.

Společnost VÍTKOVICE IT SOLUTIONS a.s. vznikla roku 25. 11. 2009 sloučením několika společností - VÍTKOVICE ITS a. s. se společností Medium SOFT a. s. a také sloučením Netprosys s.r.o., které vlastnila skupina VÍTKOVICE Machinery Group. Cílem tohoto sloučení bylo vytvoření silné IT společnosti s využitím referencí všech předchozích společností. VÍTKOVICE IT SOLUTIONS a.s. má zkušený tým profesionálů, který během malé chvíle od doby vzniku získal pro kvalitu svých služeb potřebné ISO certifikáty a osvědčení NBÚ. Společnost zaměstnává přes dvě stě zaměstnanců, z čehož jsou více jak dvě třetiny programátoři a specialisté pro počítačovou techniku [1].

Společnost se primárně zaměřuje nejen na vývoj softwaru a jeho poskytování, dodávky a implementace vlastního i partnerského softwaru v rámci skupiny VÍTKOVICE, ale také na externí trh. Náplní společnosti je systémová aplikační infrastruktura, montáž a správa počítačových sítí, serverů a doprovodné infrastruktury.

Největší firemní úspěchy, které byly vytvořeny:

- ***IS HARDIS*** – systém pro řízení výroby průmyslových podniků, nasazeno na dvou provozech VÍTKOVICE POWER ENGENEERING a.s.
- ***Implementace Siemens Teamcenter*** – nasazení, nastavení a implementace PLM systému v rámci skupiny VÍTKOVICE.
- ***IS Zaměstnanost*** – agendový informační systém pro MPSV. Evidence uchazečů o zaměstnání a související agendy.

- ***EasyGo*** – automatické odbavovací brány na letišti Václava Havla v Praze.
- ***ZZS*** – systém pro komplexní provoz záchranné zdravotní služby. Implantace ve Zlínském kraji.
- ***C3M*** – systém pro řízení krizových situací. Povodňové plány apod.

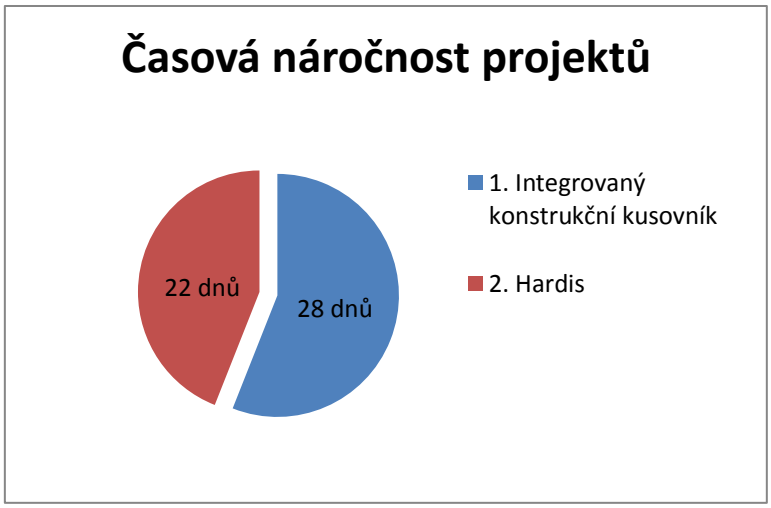
Obrázek 1: Logo společnosti

2. Projekty a zařazení studenta

Po dobu absolvování odborné praxe ve společnosti VÍTKOVICE IT SOLUTIONS a.s. jsem pracoval na dvou rozsáhlých projektech. Programování probíhalo ve vývojářském prostředí Visual Studia 2010. Mé pracovní zařazení ve společnosti bylo na postu programátora ve vývoji a analýze softwaru.

První ze dvou projektů byl Integrovaný konstrukční kusovník, na kterém jsem pracoval něco málo přes devět týdnů. Na stáž jsem dojížděl dva až tři dny v týdnu podle počtu hodin ve škole. Celkem jsem na projektu pracoval dvacet osm dnů. Pomohl jsem částečně vyřešit pár desítek problémů - jak grafického, tak i softwarového rázu.

Dále jsem pracoval na projektu Hardis. Na tomto projektu jsem odpracoval zbylých dvacet dva dnů. Pracovní náplň na projektu Hardis byla pouze softwarového druhu. Na tomto projektu mi bylo umožněno pracovat z domova pomocí VPN a vzdálené plochy, avšak této možnosti jsem využil pouze dvakrát. Hlavní obohacení na projektu bylo využití Team Foundation Serveru, se kterým jsem dosud neměl možnost pracovat. Zkušenost to byla nesmírně přínosná.



Obrázek 2: Graf rozvržených dnů

2.1 IKK – Integrovaný konstrukční kusovník

První projekt, na kterém jsem pracoval s mým konzultantem, je Integrovaný konstrukční kusovník. Konstrukční kusovník obsahuje seznam dílů potřebných k sestavení výrobku nebo jeho části. Vzniká jako podklad pro následující útvary (technologie, rozpis, nákup apod.). Je součásti výrobní dokumentace. Kusovník je možno vygenerovat i z CADu jako je SolidEdge, Inventor apod. Výstupem může být kromě tištěné podoby i strukturovaný seznam výkresů. Vytváření kusovníku v ERP systému – proto Integrovaný Kusovník.

Konstrukční kusovník je technická dokumentace součástí. Představme si například schody. Technická dokumentace má za úkol popsat, jak budou tyto schody složeny, z jakých součástí, jaké rozměry tyto součásti mají a z jakého materiálu byly vyrobeny. Proto si zaznamenáváme všechny informace do dokumentace neboli konstrukčního kusovníku. Ukázka kusovníku v příloze pod názvem A Konstrukční kusovník.

Společnost VÍTKOVICE IT SOLUTIONS a.s. má program IKK již vytvořen, ale v programovacím jazyce C++; zatímco nová modernizovaná verze bude pracovat s jazykem C#. Při mém nástupu byl projekt již z poměrně velké části rozpracován, a proto orientace v něm je velmi složitá. Nyní je zde vytvořeno něco málo přes jedenáct projektů svázaných v jednom solution. Každý z těchto menších projektů obsahuje něco málo okolo deseti tříd nebo forms. Připravené metody mi velmi pomohly při tvorbě zadaných úkolů, avšak s neznalostí velké části jsem je velice obtížně nalézal.

2.1.1 Naplnění TabGrid

Nejprve Vám zkusím přiblížit, k čemu bude sloužit plněný TabGrid a co to ten TG vlastně je. TG je jednoduchá tabulka s načtenými daty. V tomto případě se jedná o data z databáze. Celá funkčnost s naplněním bude zobrazovat vytvořené kusovníky. Zobrazené kusovníky bude možno nadále otevírat, editovat a manipulovat s nimi, ale to až v pozdějších kapitolách.

TG byl vytvořen již před mým nástupem. Jediným mým úkolem bylo vytvořit metodu, která bude načítat data z informační databáze. Vytvořil jsem proto metodu `loadBomList(Table tab)`, `tab` představuje parametr vytvořeného TG. Jediný způsob naplnění je pomocí SQL dotazu na databázi Oracle. Dotaz je i stejnojmenně uložen do místní proměnné `dotaz` typu string. Dalším krokem nastává ověření, zda byl příkaz vytvořen správně, a právě k tomu slouží podmínka `if((sql = m_sqlif.Query(dotaz)) == null)`. Metoda `Query` slouží k testování všech

vytvořených dotazů na databázi. Pokud nebude obsahovat výsledná návratová hodnota žádného výsledku, bude tedy hodnoty `null`. Znamená to, že příkaz byl napsán špatně a chybové hlášení bude vypsáno do konzole i s příslušným dotazem (Výpis 1). Jestliže bude návratová hodnota vracet kterýkoliv výsledek, bude tato podmínka ignorována.

Stačilo pouze vytvořit cyklus `while(sql.Fetch()).` Úkolem cyklu je plnit veškeré načtené informace do řádků proměnné `tab`. Plnění probíhá tak, že jsem vložil nový řádek do proměnné `tab`, následujícím řádkem kódu `Line l = tab.AddLine();`. V tuto chvíli jsem měl nový řádek v tabulce a nahrávala se získaná data z databáze příkazem `l.SetValue("PROJEKT", sql.GetString(1));`. Příkaz je použit u všech atributů, v příkladu mám uveden atribut `PROJEKT`.

```
if ((sql = m_sqlif.Query(dotaz)) == null)
{
      m_sqlif.Error("Nelze načíst položky.\n{0}", dotaz);
      return false;
}
```

Výpis 1: Ověření string dotazu

Dále však nastala mnohem větší obtíž. Jeden z atributů byl uložen jako šifrovaný a bylo potřeba jej při načtení dešifrovat. K tomuto problému jsem dostal starý dešifrovací algoritmus v jazyce C++. Stačilo ho pouze přepsat, ale vzhledem k velké obsazenosti ukazatelů a mé nezkušenosti s nimi to bylo nad mé schopnosti. Proto jsem poprosil o pomoc konzultanta společnosti, který byl tak hodný a ukázal mi, jak by tato část kódu měla vypadat. Po dokončení této části kódu již funguje načítání kusovníku do TG.

2.1.2 Otevření detailu kusovníku

Detail kusovníku pokračuje po předchozím úkolu, kdy byly načteny seznamy všech kusovníků. Při vybrání jednoho určitého kusovníku dostávám možnost otevřít detail, ve kterém jsou zobrazeny všechny podrobné informace o podsestavách a součástech. Hlavní podklady byly již vytvořeny. Opět stačilo vytvořit metodu pro plnění dat, ale v tuto chvíli se budou načítat informace o hlavičce kusovníku a následně podrobnosti do BoomGridu.

Nejprve bylo třeba vytvořit metodu pro načítání informací hlavičky kusovníku. Postup plnění probíhá stejným způsobem jako v předešlé kapitole jen se dvěma rozdíly. Nejprve jsem u vytváření dotazu na databázi využíval cyklus `for`. Důvodem byl efektivnější a hlavně rychlejší způsob programování. V minulém případě jsem musel vypisovat

jednotlivé atributy ručně, pokud jsem udělal chybu v názvu, dotaz automaticky nefungoval. Z tohoto důvodu jsem vytvořil cyklus `for`. Pomocí tohoto cyklu jsem vypisoval všechny atributy postupně - tak, jak byly definovány (Výpis 2). Ještě bylo potřeba dodělat podmínku pro identifikaci. Kusovník má specifické hodnoty `ValidTo` a `io`. Díky těmto hodnotám vyhledám pouze jedinou verzi kusovník, který chci právě načítat. Načtená data nebudou v tomto případě zobrazována v žádném Gridu, ale v textových polích.

```
string dotaz = "select\n";
for (int i = 0; i < ncitm; i++)
{
      if (i == ncitm - 1)
      {
            dotaz += itemCols[i].m_colDB + "\n";
      }
      else
      {
            dotaz += itemCols[i].m_colDB + ",\n";
      }
}
dotaz += "from\n"
+ "IIS_POLOZ\n"
+ "where\n"
+ "IO = '" + io + "'\n"
+ "AND\n"
+ "VALID_TO = '" + ValidTo + "'\n";
```

Výpis 2: Zjednodušený dotaz

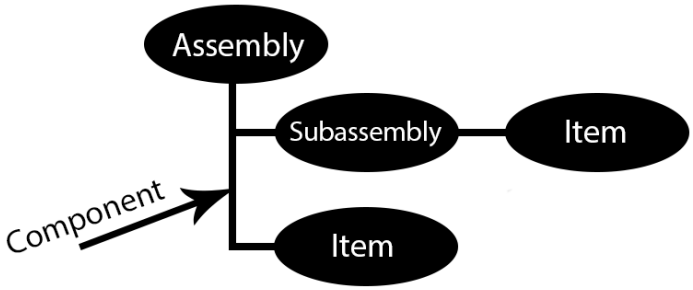
Dále bylo za úkol napsat ještě jednu metodu na právě vytvořený BoomGrid. V BG boudou zobrazeny jednotlivé součásti. Zvolím-li konstrukční kusovníku schodiště, v hlavičce budou zobrazeny informace, pod kterým číslem výkresu naleznu kusovník, kdo ho vytvořil, kdy ho vytvořil, jakou vahou součást disponuje atd., zatímco právě v BG budou vypsány informace, že schodiště bylo složeno ze zábradlí o určité délce, dále bylo složeno ze schodnic atd. Plnění je takřka totožné s předchozí kapitolou, z toho důvodu nebudu popisovat postup řešeného algoritmu.

Během práce s databází jsem byl upozorněn konzultantem firmy na neefektivitu kódu. Do této chvíle jsem velmi často načítal z databáze stejná data opětovně s velkou zátěží na databázi. Bylo mi však vysvětleno, že tato zátěž je zbytečná a mohla by být problémem, proto si potřebná data z databáze vždy před používáním načtu do místních proměnných a až následovně s nimi pracuji bez nutnosti využívání databázového spojení.

2.1.3 Nový kusovník

Vytváření nového kusovníku funguje velmi složitý způsobem. Největší obtíží bylo pochopit princip provázání jednotlivých součástí. Podklady GUI jsou totožné s načítáním detailu kusovníku jen s rozdílem, že teprve uživatel bude zapisovat informace o kusovníku.

Musel jsem poprosit konzultanta o přesné vysvětlení, jak fungují vazby v kusovníku (Obrázek 3). V kusovníku je označena hlavní sestava Assembly, která má podsestavy označeny jako Subassembly a prvky Item. Každá část kusovníku má příznaky jako prvek. Jediný způsob, jak rozlišit, o kterou část se jedná, je zjistit, jaký stav má prvek uložený například Assembly. Spojnice mezi vyšší a nižší položkou se nazývá Component. Každá má vlastní příznaky, jako například počet kusů, materiál atd.



Obrázek 3: Popis provázání kusovníku

Takže nejprve bylo potřeba vytvořit metodu a zjistit vrcholovou položku příkazem `BOMItem itm = bom.Root;`. Do místního objektu `itm` ukládám vrcholovou položku kusovníku (Výpis 3). Dalším krokem bylo uložení vrcholové položky metodou `saveItem(itm)`, kde byl sestaven SQL příkaz pro uložení.

K ukládání částí struktury jsem využíval příkaz `saveStruct(itm)`. Metoda měla za úkol uložit prvky a spojnice mezi sestavou neboli komponentu, metodou `saveComponents(asm)`. Metoda probíhala nejprve tak, že uložila všechny prvky sestavy Assembly a až následně uložila komponenty mezi sestavou a prvky. Kdybych tento průběh chtěl obrátit, nejprve uložit spojnici a až následovně prvek, neměl bych komponentu s čím svázat. Proto byl postup zvolen takto.

Po vykonání metody `saveStruct(asm)` byl vytvořen jednoduchý cyklus for pro průchod každou komponentou, která již předtím byla uložena. Cyklus měl za úkol zjistit, zda prvek, který byl svázán s touto komponentou, není podsestava (Výpis 4). Pokud zjistil, že prvek je sestava, volal rekurzivně metodu `saveStructur(itm)` s parametrem `itm`, do které právě nalezenou sestavu uložil. Cyklus probíhá do té doby, dokud neuloží všechny potřebné sestavy, komponenty a jejich prvky [2].

Tímto složitým způsobem probíhá ukládání nových kusovníků do databáze. Pochopení provázanosti všech součástí bylo velmi náročné, ale pomohlo mi udělat si širší pohled, jak celá databáze funguje.

```
bool IDIFIF.saveBom(BOMCore bom)
{
	BOMItem itm = bom.Root;

	if (!saveItem(itm))
	{
		return false;
	}

	if (!saveStruct(itm))
	{
		return false;
	}

	m_sqlif.Commit();
	return true;
}
```

Výpis 3: Metoda pro ukládání

```
private bool saveStruct(BOMItem asm)
{
	if (!saveComponents(asm)) return false;

	for (int i = 0; i < asm.Component; i++)
	{
		BOMComp comp = asm.CompByInd(i);
		BOMItem itm = comp.Item;

		if (itm.Type == eITEM_TYPE.ITEM_ASM)
		{
			if (!saveStruct(itm)) return false;
		}
	}
	return true;
}
```

Výpis 4: Rekurzivní metoda pro ukládání sestav

2.1.4 Přepočet hmotnosti

Mým úkolem bylo vymyslet algoritmus na přepočet hmotnosti sestavy. Algoritmus funguje následovně - nejprve bylo třeba zjistit, o jakou součást se jedná (plech, tyč, atd.). K tomuto poznatku jsem potřeboval zjistit, jakou hustotu materiálu má součást. Při zadávání nové součásti do kusovníku vyplňuje uživatel druh materiálu, díky tomuto jsem mohl porovnat materiál s databází, kde byly uloženy hustoty. Následně jsem jen zjistil počet kusů a vše vložil do vzorce pro výpočet. Poté ještě aktualizovat hodnotu hmotnosti v aplikaci.

Mám celkově dva způsoby, jak tento výpočet uskutečnit. První byl automatický přepočet během zadávání hodnot do seznamu prvků. Druhou možností byl ruční přepočet po stlačení ikony v menu aplikace. Tímto se dostávám k dalšímu úkolu. Měl jsem vytvořit chybějící ikony v menu. Nejprve jsem musel nakreslit ikony v malování. Následně vložit ikony do menu a aktivovat jejich funkčnost. Po zkoumání již vytvořeného menu jsem zjistil, jak vložit další. Byl to snad jeden z prvních úkolů, který se mi podařilo splnit bez zásahu konzultanta.

Metoda pro výpočet hmotnosti sestavy `compAssembly(BOMItem asm)` s parametrem představující všechny informace o sestavě (Výpis 5). Nejprve bylo potřeba zjistit počet komponent a následně procházet všechny prvky svázané na těchto komponentách. Z tohoto důvodu byl vytvořen cyklus `for`, který má za úkol zjistit kolik komponent obsahuje `comp.Quantity`. Dále počet vynásobit hmotností této komponenty a vše uložit do místní proměnné `hmot`. Podmínka if má jen úkol ověřit, jestli velikost výpočtu byla změněna od počátku. Pokud zjistí, že hmotnost je jiná, nastaví tento výpočet do objektu `asm`.

```
bool IConfig.compAssembly(BOMItem asm)
{
      int i, n;
      n = asm.Component;
      double hmot = 0.0;
      double hmot0 = asm.Hmotnost;

      for (i = 0; i < n; i++)
      {
            BOMComp comp = asm.CompByInd(i);
            BOMItem itm = comp.Item;
            hmot += comp.Quantity * itm.Hmotnost;
      }
```

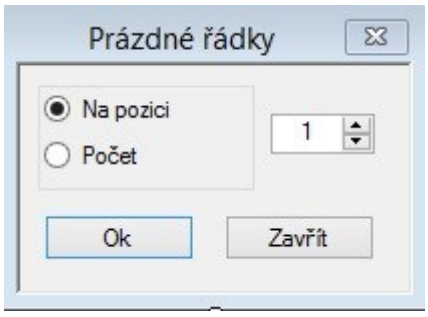
```
        if (hmot != hmot0)
        {
                asm.Hmotnost = hmot;
                asm.AddState(0, BOMItem.ITEM_HMOT);
                return true;
        }
        return false;
}
```

Výpis 5: Metoda pro výpočet hmotnosti

2.1.5 Tvorba Windows Form

Tento tvořený formulář byl pro nastavení číslování pozic. Jedná se o okraj tvorby samotné grafiky aplikace. Programátor vidí přesně to, co tvoří a ihned může posoudit výsledek, z tohoto důvodu mě odvětví nejvíce naplňuje. Nejprve jsem musel vytvořit nový soubor pro WF, do kterého bylo zapotřebí vložit labely, butony, numercUpDown a mnoho dalších prvků. Tyto nástroje bylo zapotřebí propojit s příslušnými metodami. Princip tohoto okna je na základě nastavení číslování položek v seznamu prvků. V dialogu nastaví uživatel krok, s jakým se bude číslovat a následně uloží tento parametr do algoritmu číslování [3].

Další tvorbou bylo dialogové okno (WF) pro vkládání prázdných řádků k již vytvořeným nebo právě zakládaným kusovníkům. Prvním krokem jsem musel (jako v předešlé tvorbě) navrhnout grafické rozmístění všech nástrojů a poté jsem mohl vytvářet funkce (Obrázek 4). Dialog pro vkládání prázdných řádků byl rozdělen na dvě části. Uživatel chce vkládat prázdné řádky na zvolenou pozici před spuštěním dialogu, který jsem právě vytvářel. Druhou možností byl způsob vkládání na uživatelovu zvolenou pozici přímo v dialogu.



Obrázek 4: Dialog prázdné řádky

Mám-li například spuštěný dialog, zvolil jsem pomocí CheckButtonu možnost pro vkládání na mnou zvolený řádek. Zvolil jsem si řádek s číslem pět a poté algoritmus vyplní řádek. V druhém případě zvolím, že chci vkládat prázdné řádky od poslední známé pozice v seznamu. Pak už jen vyplním počet řádků. Pouze pro možnost s volbou počet můžu vkládat více než jeden řádek. Dále jsem doplnil další kontrolní dialogové okno, zda uživatel opravdu chce vložit prázdné řádky.

K samotnému algoritmu - bylo potřeba volat metodu, která byla již vytvořena pro získání vybrané pozice v seznamu, a od té se budou pouze vkládat prázdné řádky. Všechny následující řádky budou posunuty s nastaveným krokem o počet vložených řádků. Konzultantem mi byly vysvětleny možnosti, jak zbytečně nevytvářet nové kontrolní dialogové okno s potvrzení změn. Tento dialog je již součástí vytvořeného rozhraní vitsgui v projektu. Pomocí příkazu `MBox.Warning2` můžu vyvolat dialogové okno s parametry pro nadpis okna, text a index u příkazu Warning mi určoval, o jaký druh dialogu se jednalo. Mnou zvolený byl se dvěma tlačítky.

Podle zvolené odpovědi bylo možné odchytávat, o jakou odpověď se jednalo a podle té pak rozhodovat, co se bude dále kód vykonávat. V mém případě jsem varoval uživatele před tím, že nyní bude docházet k posunutí pozic po následku vkládání nových prázdných řádků (Výpis 6). Pokud uživatel bude souhlasit, příkaz `DialogResult.Yes`, kód bude pokračovat a všechny pozice za nově vloženým řádkem budou posunuty. V opačném případě, když uživatel s tímto nebude souhlasit, se nevykoná nic [4].

```
if (!itm.CanInsert(first_poz, last_poz, out first_move))
{
      if (MBox.Warning2("Posunutí!", string.Format(
            "Pro vložení ……?")) == DialogResult.Yes)
      {
            itm.Odsun(first_poz, last_poz, (int)Step);
            refreshAsmPos(itm);
            m_cmd_save.Enable(true);
            m_cmd_copyAsm.Enable(false);
      }
      else
      {
            return;
      }
}
```

Výpis 6: Integrované dialogové okno

Ještě bylo potřeba vytvořit funkčnost pro automatické vkládání prázdných řádků. Problematika byla následující - když uživatel smaže jakýkoliv prvek v kusovníku, dojde k tomu, že se v seznamu vyskytne prázdné místo. Mnou vytvořená metoda má za úkol toto prázdné místo vyplnit. Podmínka zněla tak, že jestliže bude počet těchto mezer větší než čtyři, nemůže dojít k doplnění. Z tohoto důvodu bylo zapotřebí nejprve spočítat, kolik prázdných míst obsahuje kusovník. Potřeboval jsem for cyklus, který by umožnil průchod celým seznamem prvků s porovnáváním aktuálního prvku s pozicí a následný prvek s pozicí. Pokud metoda nalezne rozdíl v umístění obou prvků neboli mezery, dále pouze zjistí, kolik mezer zde bylo vytvořeno. Po průchodu si ověříme, že číslo nepřesahuje podmínku maximálně čtyř mezer a pokud ne, vyplníme mezery prázdnými řádky.

2.1.6 Kopírování sestav

Funkčnost metody byla velmi komplikovaná. Musel jsem vytvořit kopii již uložené sestavy. Jednalo se tedy o kopírování informací do nového kusovníku. Nejprve však bylo potřeba vytvořit potřebné grafické podklady, tlačítka, ikony a další nástroje. Po tvorbě grafických úkonů přicházela na řadu tvorba metody.

Hlavním krokem bylo především zjistit, který řádek byl uživatelem označen v seznamu součástí kusovníku. Pomocí řádku jsem zjistil prvek pro kopírování, tedy ID a revizi prvku. Díky těmto poznatků jsem mohl vytvořit kopii v zásobníku. Zásobníkem bylo myšleno dialogové okno, kde se uchovávají veškeré kopírované nebo vyjmuté prvky či sestavy, které jsem následně vkládal do nově vytvořeného nebo editovaného kusovníku. Celkem byly dva způsoby jak vložit prvek do zásobníku.

První způsob byl klasicky kopírováním a druhý vyjmutím. Jestliže uživatel zvolí možnost prvek vyjmout z původního kusovníku, dojde k smazání originálu a prvek bude pouze v zásobníku. U vyjmutí bylo dále důležité kontrolovat, jestli není kusovník uzamknut nebo vytvořen jiným uživatelem. A pokud ano, přístup k metodě pro bude zamítnut. Každý uživatel má práva jen k tomu, co sám vytvářel. Poté už jen stačilo z těchto získaných poznatků v novém kusovníku volat informace ze zásobníku, ale bez revize a ID, ty se budou vytvářet znova, aby nedocházelo k návaznosti. Pokud by tato návaznost přetrvávala, při jakémkoli mazání nebo editaci by se projevila změna v obou kusovnících. Dále stačí kusovník už jen uložit, a to pomocí vytvořené metody, kterou jsem již dělal v předchozí kapitole.

2.1.7 Uzamknutí sestavy

Uzamykání a odemykání sestav pro ochranu před jejich nechtěnou editací od jiné osoby než je majitel vytvořeného kusovníku. Jako vždy bylo potřeba vytvořit veškeré grafické podklady. Při zadávání příkazu uzamykání bylo nezbytně důležité zjistit, zda samotná sestava již nebyla někým uzamčena a to tak, že si načtu hodnotu `locker` z databáze. Jestliže se `locker` rovnala prázdné hodnotě string nebo je nastavena na hodnotu null, bylo patrné, že kusovník ještě nikdo neuzamknul a proto v této části algoritmu nastavím hodnotu `locker` na aktuálně přihlášeného uživatele. Nastane-li případ, že hodnota `locker` byla shodná s proměnnou `m_Name`, do které jsem uložil informaci o přihlášeném uživateli (Výpis 7). Vyvolal jsem výjimku pro dialogové okno „Sestavu již máte uzamčenou.“. Poslední možnost, ke které může dojít. Načtená hodnota `locker` nebyla rovna přihlášenému uživateli a v tuto chvíli dojde opět k výjimce s informativním dialogem.

```
if (locker == m_Name)
{
      m_err_msg = "Sestavu již máte uzamčenou.";
      m_sqlif.Rollback();
      return false;
}
m_err_msg = "Sestava je uzamčená uživatelem " + itm.Locker +
     ", nelze provádět změny.";

m_sqlif.Rollback();
return false;
```

Výpis 7: Podmínka uzamknutí

Při úspěšném zamykání jsem nastavoval hodnotu proměnné na `readOnly = 0`, aby nebylo možno dále manipulovat s daty. Odemykání probíhalo však mnohem snadněji. Algoritmus opět zjistil, kdo kusovník uzamknul. Pokud docházelo k situaci, že kusovník uzamknul někdo jiný než přihlášená osoba v systému, zobrazovalo se pouze dialogové okno s varováním. V opačném případě však kusovník uzamknul uživatel, který je právě přihlášen, nastaví se hodnota `locker = null` a `readOnly = 1`. Tímto jednoduchým způsobem mohl uživatel zrušit uzamknutí kusovníku. V poslední řadě probíhá ukládání všech změn do databáze.

2.1.8 Překladový slovník

Slovník v projektu IKK bude sloužit k uchování překladu český výrazů do jiných jazyků. Bylo potřeba vytvořit nové dialogové okno, ve kterém bude DropDown list pro možnost vybrání jazyků. DropDown list bude načítat všechny uložené druhy jazyků z databáze. Pro zobrazení samotných překladů jsem vytvořil DataGridView. Samotné plnění DGV nebylo nic složitého. Jednoduchý dotaz na příslušnou relaci v databázi a pouhé plnění řádků DGV.

Hlavní nejtěžší funkcí dialogu bylo ukládání a načítání ze souboru typu Excel. Do této chvíle jsem se s ničím podobným ještě nesetkal. Naštěstí celá tato metoda byla již vytvořena v předchozí verzi programu a byla převedena do nové verze. Z tohoto důvodu metodu stačilo pouze zavolat na mnou vytvořený DGV. Nutné bylo vyřešit ukládání a mazání slovních překladů. Ukládání nebylo možné provádět standartním způsobem. Slovíčka neměla totiž svá identifikační čísla, podle kterých by bylo možné editovat nebo mazat či přidávat nová. Implementace metody právě proto probíhala nejprve mazáním všech původních slovíček a následně vše z DGV uložila. Způsob implementace není podle mého úsudku korektní, ale zásahy do tabulek databáze mi nebyly umožněny. Vzniká zde velká zátěž na databázi. Je rozdíl, pokud do databáze ukládám pět nových slovíček nebo opětovně ukládám všechny záznamy - například se sto řádky, které ještě před uložením musím smazat.

2.1.9 Překlad frází a jazykové mutace

Překlad frází v kusovníku fungoval na následujícím principu. V menu uživatel vyvolá nové dialogové okno, ve kterém byl seznam všech jazyků (od angličtiny až, dejme tomu, po ruštinu). Po vybrání jazyka přichází ověření, zda uživatelem zvolený překlad již někdy v minulosti neuložil pro právě aktuální kusovník. První alternativa programu zjistí, že kusovník ještě nikdy nebyl přeložen a vyzve uživatele k překladu frází podle slovníku. Pokud však dojde k druhé alternativě, že uživatel již slovník uložil v minulosti, program pouze zobrazí uložený překlad neboli jazykovou mutaci.

Samotný překlad slovníku byl formou dotazu na databázi (Výpis 8). Dotaz byl řízen podmínkou s parametrem `OraPar.PackString(frase.Substring(sti, flen))`, který představoval jedno slovo fráze a tento parametr porovnával s českým výrazem v databázi. Druhá část podmínky bylo porovnání identifikačního čísla jazyka s databází `JAZYK = lang`. Proměnná lang přestavuje ID vybraného druhu jazyka.

```
string dotaz = "select PREKLAD from IIS_SLOV\n"
      + "where \n"
      + "JAZYK = " + lang + "\n"
      + "and CESKY = " +
           OraPar.PackString(frase.Substring(sti, flen));

if ((sql = m_sqlif.Query(dotaz)) == null)
{
      m_sqlif.Error("Cannot run query\n{0}", dotaz);
}

if (sql != null && sql.Fetch())
{
      pr = sql.GetString(0);

      if (cp == "1")
      {
            pr = OraPar.LatToAzb(pr);
      }
      sql.Close();
      break;
}
```

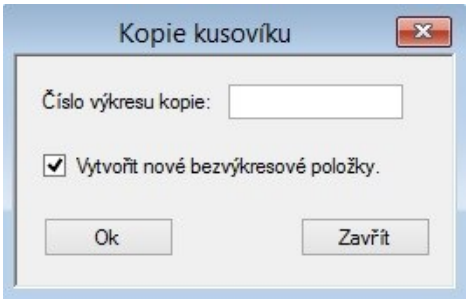
Výpis 8: Vyhledání fráze v databázi

Důležitým prvkem pro překlad těchto frází je zjištění, zda se nejedná o azbuku. Pokud je jazyk v azbuce `if (cp == "1")`, program zavolá metodu pro převod písmen do azbuky nebo naopak `LatToAzb(pr);`. Tímto způsobem můžeme přeložit základní fráze v kusovníku podle slovníku, který jsme vytvářeli v předchozí kapitole.

Překlady se daly ukládat do databáze ke každému z kusovníků. Slovník totiž nemusel obsahovat všechny slova obsažená v kusovníku, proto si může uživatel slova sám přeložit a uložit zvlášť pouze pro kusovník. Princip ukládání fungoval jako u kapitoly, kde jsem ukládal nově vytvořené kusovníky jen s tím rozdílem, že v předešlé části byl staticky vyplněn parametr pro druh jazyka na jedničku. Jednička při typu jazyka znamenala český jazyk. Po uložení se mi již slova pokaždé zobrazovala přeložená, pokud si zvolím překlad právě do daného jazyka.

2.1.10 Kopie kusovníku

Kopie kusovníku byla velmi podobná kopírování sestavy v předchozí kapitole, jen s tím rozdílem, že v tuto chvíli budu kopírovat celý kusovník bez žádné změny, pouze bude nově pojmenován. V menu byla vytvořena nová položka a do ní jsem vložil nový dialog, ve kterém bylo textové pole nové pojmenování výkresu. Dále pak checkbox s informací, zda má být tvořen kusovník s bezvýkresovými položkami (Obrázek 5).



Obrázek 5: Dialog kopie kusovníku

Celá aktuální sestava je uložena v jedné proměnné a obsahuje vždy aktuální informace o kusovníku. Tyto informace upravím tak, že změním jméno výkresu, resetuji stavy vše ID, aby mohl být kusovník znovu vložen pod jiným ID a nestalo se, že se bude provádět pouze update. Dále je třeba ještě ošetřit ukládání informací pro jazykové mutace v kopii kusovníku. Tuto problematiku ale nechám na později. Ukládání se mi zdařilo úspěšně vyřešit, ale dlouho jsem nevěděl, jak udělat, aby se mi otevřel nový kusovník a originální zůstal nezměněn. Jde o to, že upravuji data v originálním kusovníku, tím pádem se mi mění i v zobrazeném dialogu. Vyřešil jsem problém tak, že jsem si zapamatoval ID originálu, nově vytvořeného a následně oba kusovníky otevřel a předlohu zavřel. Nevím, zda je toto řešení korektně udělané, ale je funkční. Dnes jsem se dozvěděl, že od příštího týdne budu pracovat na jiném projektu, s kterým chtějí pomoci, proto tato problematika není odladěná a dořešená.

2.2 Hardis

Projekt slouží pro řízení výroby průmyslových podniků, nasazeno na provoze VÍTKOVICE POWER ENGENEERING a.s. Systém Hardis dokáže pokrýt oblasti všech činností společnosti spojených s výrobou. Hlavní prioritou jsou však technické dokumentace zakázek, odvádění výroby a plánování. Hardis obsahuje také pomoc při sledování mzdových nákladů, kontrolu plnění výrobních plánu a norem jak pro střediska pracovišť, ale také u jednotlivců. Je to hlavní řídící jednotka pro řízení výroby [5].

Hlavní funkcí systému jsou detailní informace zakázek, kusovníky výrobků, výrobní plány, úkolové lístky, reporting. Do výstupu ze systému dále náleží technologické postupy, normy časů, kapacitní plány, úkolové lístky, reporty, expediční lístky, atd.

Projekt Hardis je mnohem rozsáhlejší než je projekt IKK. Dalo by se říct, že projekt IKK, který jsme popisovali v předchozí části, je jen jedna malá část v tomto rozsáhlém projektu. K projektu Hardis jsem se dostal hlavně z důvodu výpomoci a časového presu spolupracovníků, z důvodu již funkčního projektu. Během testování a ladění projektu dochází k odhalení chyb, také následným požadavkům na zlepšení systému. Jde o to, že na projektu pracuje přibližně deset lidí. Jen na pobočce, kde jsem pracoval i já, pracují na projektu čtyři lidé a mimo jiné také pan Chamrád, můj konzultant. Tým všech vývojářů právě proto využívá pro práci s jedním projektem funkci Team Foundation server [6].

Seznámení s projektem bylo velmi složité. Byl to neuvěřitelně rozsáhlý projekt, který byl již dávno zprovozněn na provoze. Proto byl nápor na psychiku mnohem větší. Naštěstí vše pracuje na dvou databázích, testovací a ostré verzi.

2.2.1 Přepočty normo časů

První a jediný úkol, který mi byl v projektu Hardis zadán, byly přepočty normovacích časů. Normo časy v systému Hardis slouží mistrům k výpočtu časů pro pracovníka nebo pracovní skupiny ke splnění celého pracovního úkonu. Měl-li mistr pracovníka a tomu zadal úkol brousit tří metrový nosník o šířce třiceti centimetrů, algoritmus vypočítal, jaká průměrná doba bude potřeba, aby pracovník splnil zadaný úkon. Přepočty normovacích časů jsou rozděleny do části pro jednotlivé pracoviště - například svářeč, svářeč CO, zámečník atd.

Mým úkolem bylo upravovat tyto algoritmy podle požadavků mistrů. Dostával jsem zadání, kde byly vypsány jednotlivé úkony a požadované změny nebo pokyny k tvorbě úplně

nových činností. Grafický zásah do částí kódu nebyl vůbec zapotřebí. Všechny způsoby vytváření DGV byly automatické v samotném grafickém rozhraní programu. Sloupce v DGV byly tvořeny podle počtů prvků v databázi. Způsob tvorby mě velmi fascinoval. Velmi zjednodušuje a hlavně šetří práci programátora.

Orientace v projektu Hardis byla velmi složitá. Rozsáhlost je pro mne až neuvěřitelná. Prvotní úpravy byly řešeny pouze podle zadání a více jsem nedokázal nahlédnout do problematiky. Po uplynutí času, kdy mé programovací znalosti začaly pomalu nabývat, jsem začal přicházet na to, jak zefektivnit části kódu, které psal někdo přede mnou. Hlavní nároky na úpravu byly dělat kód co nejvíce efektivním, bez budoucího zásahu programátora. Z tohoto důvodu byly konstanty časů ukládány do databáze a zobrazeny v jiné části grafického rozhraní Hardisu. Nejlepší způsob bude ukázat si vše na příkladu.

Technolog bude vytvářet zámečníkovi operaci přípravu plechů pro pásnice a mistr mu ji zadá. Činnost obsahuje celkem tři parametry, a to délku, šířku a ohyb. V databázi mám vytvořenou tabulku s časy. Ve sloupcích byly definovány délky a v řádcích šířky (Obrázek 6). Parametr ohybu mi celkový čas zvyšoval o dvacet procent. Mistr zvolil délku dva a půl metru se šířkou půl metru. Plech se bude ohýbat.

Koeficienty - PLECHY

Délka[mm] / Šířka[mm]	100	200	300	400	500	600	8
1000	8	8	8	8	8	12	1
2000	9	10	11	12	12	13	1
3000	12	15	17	20	22	24	3
4000	15	17	20	22	24	27	3

Obrázek 6: Tabulka koeficientů

Algoritmus získá zvolené hodnoty a pomocí nich zjistí, který sloupec nebo řádek vyhovoval zadání. K získání indexu řádku právě slouží mnou vytvořená metoda `GetRow(delka, table, 0);` (Výpis 9). Parametry v metodě označovaly `delka` zvolenou délku, tedy v mém případě dva a půl metru. Další `table` určuje načtenou tabulku z databáze do dvourozměrného pole typu double. Poslední parametr mi naznačoval, od kterého řádku začne metoda porovnávat hodnoty. Nejdůležitější částí metody byl cyklus for, přitom zároveň nejjednodušší. Pouze měl za úkol spočítat počet řádků a porovnávat

hodnoty zadané s hodnotami v tabulce. Pokud podmínka `if (table[i, index] >= hodnota)` byla splněna, hodnota s délkou byla nalezena. Celá metoda vrací index řádku.

```
public static int GetRow(double hodnota, double[,] table, int
index)
{
      if (table == null)
         return -1;

      for (int i = 0; i < table.GetLength(0); i++)
      {
            if (table[i, index] >= hodnota)
            return i;
      }
      return -1;
}
```

Výpis 9: Metoda pro vyhledání řádku

Stejný způsob fungoval pro metodu vyhledávání sloupce. Jen s rozdílem, že se porovnávala hodnota pro zadanou šířku. Po vyhledání obou indexu mi vznikl průnik v tabulce s časovou hodnotou. Nalezený čas jsem už pouze vynásobil dvaceti procenty, protože mistr zvolil ohyb plechu. Výsledný čas jsem zobrazil mistrovi.

Metod typu GetRow jsem vytvářel přibližně deset. V předchozím případě bylo porovnáváno číslo. V některých ale byly porovnávány hodnoty string. Jedny z nejsložitějších metod byly vytvářeny porovnání stringů, ve kterých byly uloženy dva parametry. Musel jsem využívat možnost rozdělit stringy do pole pomocí příkazu split. Pak už stačilo vytvořit dvě podmínky if pro každý string [7].

Přibližně mnou bylo upraveno nebo vytvořeno kolem čtyřiceti metod tohoto typu. Mnohdy však mnohem složitějších, kde bylo potřeba získat až šest indexů v tabulkách, zatímco v ukázkovém příkladu pouze dva. Vzhledem k opakovanosti vyhledávání jsem byl nucen vytvářet univerzální metody pro vyhledávání v tabulkách. Od té chvíle programátor zavolá jen příslušnou metodu, která mu vrátí index v tabulce.

3. Uplatněné znalosti

Implementace v jazyce C# byla hojně využívána v obou projektech. S jazykem C# jsem se setkal při studiu na vysoké škole v předmětech *Programovací jazyky II*, *Databázové a informační systémy* a v poslední řadě v předmětu *Vývoj informačních systémů.* Díky vysoké škole jsem měl základní přehled implementace. Praxe však proti teoretickým znalostem byla velmi odlišná. Díky možnosti absolvování praxe se základní implementací nyní nemám potíže. Zjistil jsem, že jakýkoliv problém se dá rozdělit do menších celků a ty se dají vyřešit.

Zastoupení databázových systémů je v dnešní době nedílnou součástí každého systému s potřebou ukládání dat pro širší použití. Ve společnosti VÍTKOVICE IT SOLUTIONS a. s. najdeme zastoupení ve verzi SQL pro Oracle. S databázovými systémy jsem se také setkal při studiu na vysoké škole. Zastoupení měly v předmětech *Úvod do teoretické informatiky* a *Databázové a informační systémy*. Průprava těmito předměty byla více než dostatečná na všechny zadané úkoly tohoto typu.

Team Foundation Server byl pro mě něčím úplně novým. Jedná se o možnost spolupráce více lidí na jednom projektu bez možnosti kolizí. Kolize mohou nastat v případě, pokud v jednom projektu pracují dva lidé na jedné části. Právě díky TFS k podobným událostem nedochází. Pokud programátor chce pracovat na určité problematice, uzamkne část, na které bude pracovat a tím kolizi zabrání. Mezitím jiný programátor může pracovat na jiné části. TFT je určitě nedílnou součástí velkých projektů, z tohoto důvodu jsem velmi vděčný za možnost seznámení s tímto způsobem implementace. Během studia na vysoké škole nebyla možnost požití TFS.

4. Závěr

Bakalářská práce formou absolvování odborné praxe mne velmi zaujala a zdála se mi jako dobrá příležitost získat nové zkušenosti v oboru, který studuji. Mé předpoklady se po absolvování praxe také vyplnily. Výhodou odborné praxe je čerpání zkušeností z každodenního chodu firmy zabývající se vývojem softwaru. Tyto zkušenosti budou velmi potřebné po dokončení studia. Také velkým přínosem pro mne, jako studenta, je ověření si svých nabytých vědomosti získaných během studia na Vysoké škole báňské – Technické univerzitě v Ostravě.

Praxe v externí firmě byla pro mě velkým přínosem. Již v průběhu absolvování odborné praxe mi zkušenosti z řešení úkolů pomohly při studiu. Získal jsem větší přehled o tom, jak implementovat programovací jazyky a to nejen v jazyce C#. Dřívější základní potíže s implementací již pro mne v tuto chvíli nehrají roli.

Všechny mnou vytvořené části algoritmů budou použity při dalších postupech vyvíjení softwaru. Tohle budu považovat za poctu, jelikož jsem mohl přispět mými znalostmi pro společnost VÍTKOVICE IT SOLUTINS a.s. Z důvodu velké obsáhlosti projektu konstrukčního kusovníku nebylo možné vše dokončit před ukončením stáže. Finální verze projektu je odhadována ke konci roku 2014. Zatím co u již funkčního projektu Hardis jsem vypomáhal pouze s laděním algoritmů, z tohoto důvodu mi byly ihned referovány reakce zaměstnanců, kteří s upravenou částí pracovali. Přínosem pro mě také bylo využívání Team Foundation Serveru, se kterým bych se při studiu na vysoké škole nesetkal v takovém měřítku.

Použitá literatura

[1] *VÍTKOVICE IT SOLUTIONS a. s.: O společnosti* [online]. © 2011 [cit. 2014-04-21]. Dostupné z: http://itsolutions.vitkovice.cz/

[2] KHORSHINIA, Shahin. *CodeProject - For those who code: Recursive methods using C#* [online]. 23. 4. 2013 [cit. 2013-04-22]. Dostupné z: http://www.codeproject.com/Articles/142292/Recursive-methods-in-Csharp

[3] *Microsoft Developer Network: Form Class* [online]. © 2014 [cit. 2013-12-12]. Dostupné z: http://msdn.microsoft.com/en-us/library/System.Windows.Forms.Form(v=vs.110).aspx

[4] SHARP, John. *Microsoft Visual C# krok za krokem 2010*. Brno: Computer Press, 2012. ISBN 978-80-251-3147-3.

[5] *VÍTKOVICE IT SOLUTIONS a. s.: Hardis* [online]. © 2011 [cit. 2014-04-21]. Dostupné z: http://itsolutions.vitkovice.cz/37/cs/node/2065

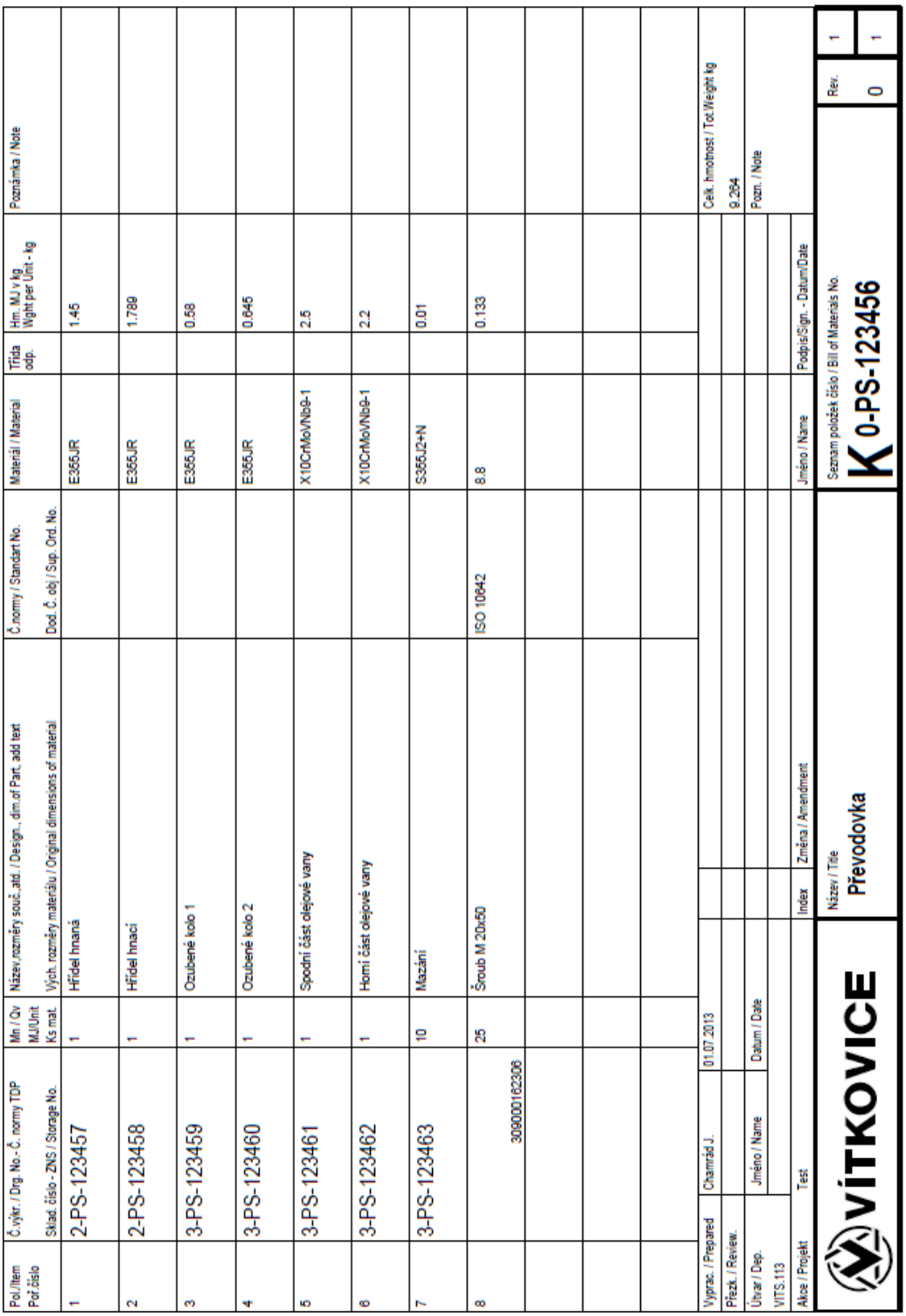
[6] *Microsoft Developer Network: Team Foundation Server* [online]. © 2014 [cit. 2014-02-01]. Dostupné z: http://msdn.microsoft.com/en-us/vstudio/ff637362.aspx

[7] *Microsoft Developer Network: String.Split Method(Char[])* [online]. © 2014 [cit. 2014-06-01]. Dostupné z: http://msdn.microsoft.com/en-us/library/b873y76a(v=vs.110).aspx

Přílohy

A Konstrukční kusovník

Uložen v přiloženém CD pod názvem „A Konstrukcni kusovnik“.



Pol./Item Poř.číslo	Č.výkr. / Drg. No.- Č. normy TDP Sklad. číslo - ZNS / Storage No.	Mn / Qv MJ/Unit Ks mat.	Název,rozměry souč.,atd. / Design., dim.of Part, add text Vých. rozměry materiálu / Original dimensions of material	Č.normy / Standart No. Dod. Č. obj / Sup. Ord. No.	Materiál / Material	Třída odp.	Hm. MJ v kg Wght per Unit - kg	Poznámka / Note
1	2-PS-123457	1	Hřídel hnaná		E355JR		1.45	
2	2-PS-123458	1	Hřídel hnací		E355JR		1.789	
3	3-PS-123459	1	Ozubené kolo 1		E355JR		0.58	
4	3-PS-123460	1	Ozubené kolo 2		E355JR		0.645	
5	3-PS-123461	1	Spodní část olejové vany		X10CrMoVNb9-1		2.5	
6	3-PS-123462	1	Horní část olejové vany		X10CrMoVNb9-1		2.2	
7	3-PS-123463	10	Mazání		S355J2+N		0.01	
8	3090001622306	25	Šroub M 20x50	ISO 10642	8.8		0.133	

Celk. hmotnost / Tot.Weight kg: 9.264

Vyprac. / Prepared: Chamrád J. — 01.07.2013
Přezk. / Review:
Uhyar / Dep.: Jméno / Name — Datum / Date
VITS.113
Akce / Projekt: Test — Index — Změna / Amendment — Jméno / Name — Podpis/Sign. - Datum/Date

Pozn. / Note

VÍTKOVICE

Název / Title: Převodovka

Seznam položek číslo / Bill of Materials No.: K 0-PS-123456

Rev.: 0

1 / 1